Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Haris Alijagić

Grafični uporabniški vmesnik za pomoč pri prevajanju spletnih aplikacij

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Luka Šajn

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Izdelajte grafični uporabniški vmesnik, ki bo preprost za uporabo in v pomoč pri prevajanju in urejanju besedil v spletnih aplikacijah. Grafični vmesnik naj komunicira s strežnikom, ki hrani podatke o prevodih. Komunikacija naj poteka preko REST API vmesnika strežnika. Naj bo dovolj splošen in konfigurabilen, tako da se lahko preprosto vključi v katero koli spletno aplikacijo. Izgled vmesnika naj bo ena izmed glavnih prioritet in uporaba naj bo preprosta, saj je cilj, da lahko aplikacije prevajajo tudi tehnično neizkušene osebe ali stranke, ki so kupile neko spletno aplikacijo kot produkt in želijo spreminjati prevode.

Zahvaljujem se zaposlenim v Halcomu za idejo in nasvete pri tehničnih težavah, mentorju za pomoč pri pisanju, ter družini in prijateljem za podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
REST	representational state transfer	predstavitveni prenos stanj
API	application program interface	vmesnik uporabniškega programa
CORS	cross-origin resource sharing	
GUI	graphical user interface	grafični uporabniški vmesnik
EAP	enterprise application platform	aplikacijska platforma za podjetja
CDI	contexts and dependency injection	vstavljanje vsebini in odvisnosti
HTTP	hypertext transfer protocol	protokol za prenos podatkov preko spleta
HTTPS	hypertext transfer protocol – secure	varen protokol za prenos podatkov preko spleta
URL	uniform resource locator	enolični krajevnik vira
HTML	hypertext markup language	označevalni jezik za izdelavo spletnih strani
CSS	cascading style sheets	kaskadne stilske predloge

Povzetek

Naslov: Grafični uporabniški vmesnik za pomoč pri prevajanju spletnih aplikacij

Avtor: Haris Alijagić

Napačni prevodi in druge jezikovne nepravilnosti znotraj aplikacije odražajo slabo kvaliteto in mečejo slabo luč na podjetje, ki je aplikacijo razvilo. Poleg tega lahko privede tudi do zmede pri uporabnikih. Vse to na koncu doprinese k slabemu ugledu podjejta in k izgubi dobička. Zato je postalo prevajanje nujen del razvojnega procesa. Uporabnik ne potrebuje posebnega računalniškega znanja, da lahko začne prevajati, temveč mora biti dovolj preprosto za vsakogar. Rezultat diplomske naloge je preprost in intuitiven GUI, ki se ga lahko enostavno vključi v katero koli spletno aplikacijo in začne s prevajanjem.

Ključne besede: prevodi, razvojni proces, spletna aplikacija, grafični uporabniški vmesnik.

Abstract

Title: Graphical user interface for translating web applications

Author: Haris Alijagić

Wrong translations and other grammatical errors in an application suggest bad quality and cast a bad light on the company that developed the application. It can also cause confusion with the users. This can foster a negative impression of the company, resulting in a bad reputation and loss of profit. As a result, translating has become an important part in the development process of an application. A user should not need advanced technological skills to be able to start translating an application, it must be simple enough for anyone. This thesis provides a simple and intuitive graphical user interface, that can be easily included into any web application and used for translating.

Keywords: translations, development process, web application, graphical user interface.

Poglavje 1

Uvod

Aplikacije so orodja, ki jih uporabljamo, da opravimo neko nalogo. Da lahko ta orodja uporabljamo, mora biti njihova uporaba razumljiva. Ena izmed najpogosteje uporabljenih tehnik za predstavitev informacij na preprost in razumljiv način je uporaba besedila. Število aplikacij, zlasti spletnih aplikacij zelo hitro narašča in s tem raste tudi potreba po prevajanju. Ker je prevajanje postalo ključen del razvojnega procesa, sem izdelal orodje namenjeno spletnim aplikacijam, ki ta proces poenostavi in s tem tudi pohitri. Velika prednost orodja je tudi to, da ga lahko uporablja kdorkoli. V praksi to pomeni, da se razvojnikom ni potrebno obremenjevati s prevodi, ampak lahko to delo opravi tudi nekdo drug.

Poglavje 2

Raziskovanje obstoječih orodij za pomoč pri prevajanju

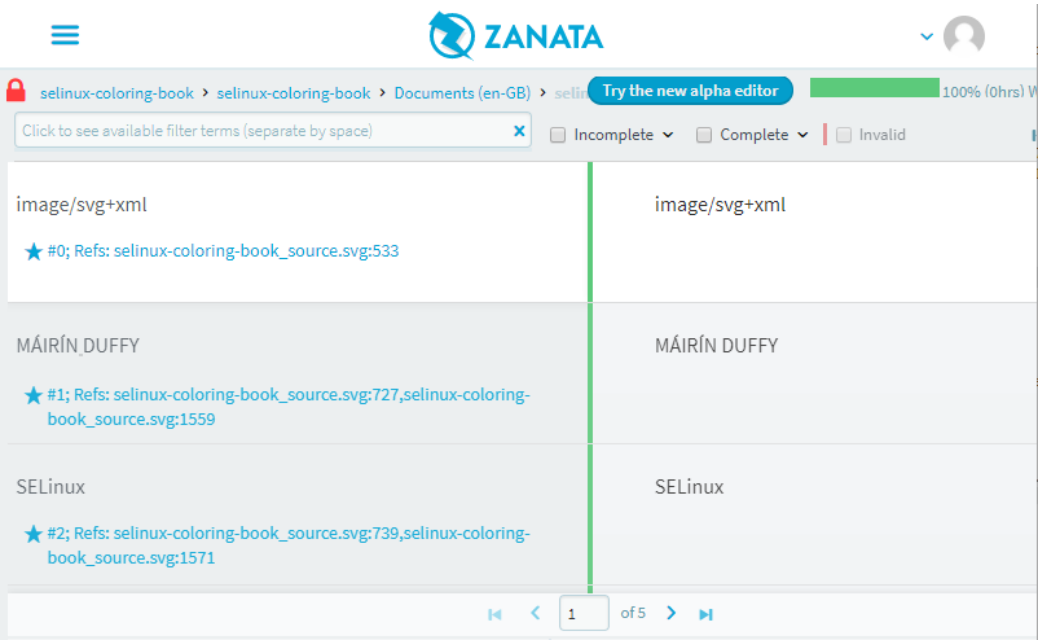
Pred pričetkom razvijanja rešitve, sem raziskal že obstoječa orodja in tehnologije, ki jih lahko uporabim pri razvoju svoje aplikacije. Ker sem problem prevajanja omejil na spletne aplikacije, sem seveda uporabil spletne tehnologije kot so HTML, CSS in JavaScript. Uporabil sem tudi JavaScript knjižnico jQuery, ki zelo poenostavi razvijanje v JavaScriptu. V nadaljevanju predstavljam ključna orodja, ki sem jih uporabil za razvoj aplikacije.

2.1 Zanata

Zanata je odprtokodna spletna aplikacija namenjena upravljanju s prevodi. Razvilo jo je podjetje Red Hat. Napisana je v Javi in uporablja moderne tehnologije kot so JBoss EAP, CDI, Hibernate in REST API [12]. Podjetje v katerem sem zaposlen, se je za uporabo orodja Zanata odločilo zaradi naslednjih ključnih točk:

- je odprtokodno orodje
- ima enostaven dostop preko brskalnika
- več ljudi lahko prevaja sočasno

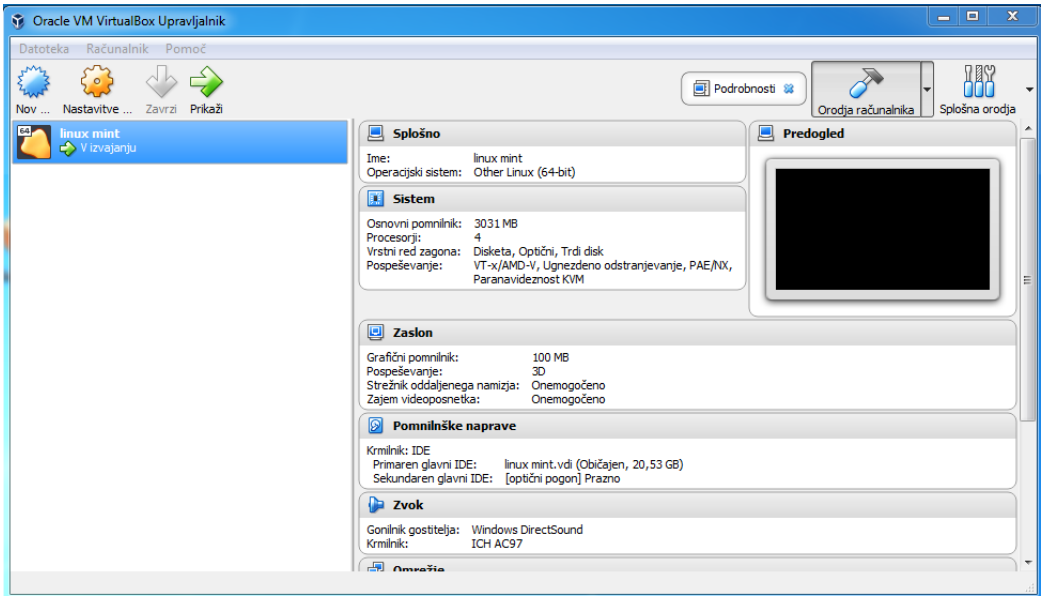
- omogoča več različic prevodov
- izvorna koda je napisana v Javi
- ima REST API



Slika 2.1: Spletna aplikacija Zanata [10]

2.2 Virtualbox

Virtualbox je zastonjski odprtokodni program, s katerim lahko poganjamo navidezne računalnike z različnimi operacijskimi sistemi znotraj našega glavnega operacijskega sistema. Razvilo ga je podjetje Innotek GmbH, kasneje ga je prevzelo podjetje Sun Microsystems, ki je med drugim razvilo tudi programski jezik Java, po pridobitvi podjetja Sun Microsystems s strani podjetja Oracle pa si ga lasti le ta. Ker je strežnik Zanata najenostavneje postaviti na operacijskem sistemu Linux, sem uporabil Virtualbox, da sem lahko poganjal Linux na svojem računalniku z operacijskim sistemom Windows [9].



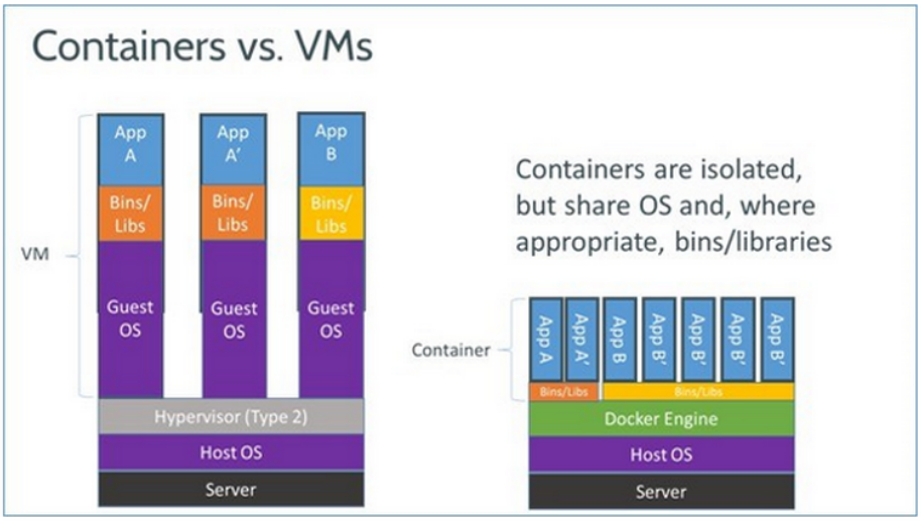
Slika 2.2: Program Virtualbox

2.3 Docker

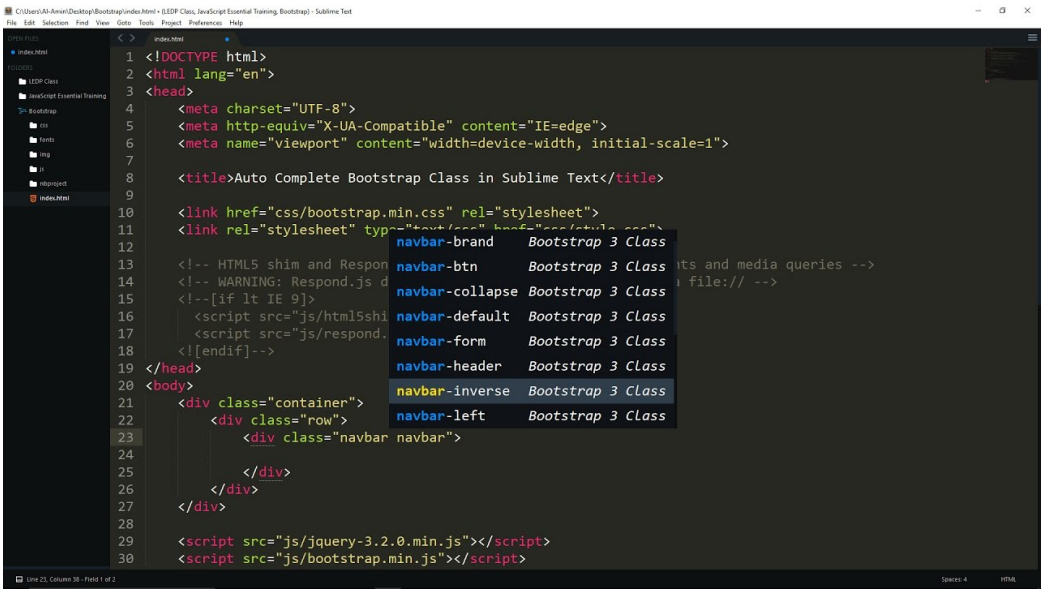
Docker je tehnologija, ki uporablja zmožnost Linux jedra, t. i. izolacijo virov in nam s tem omogoča, da poganjamo več neodvisnih vsebnikov znotraj enega primerka Linuxa. Docker vsebniki tečejo na gostujočem operacijskem sistemu, kar primore k manjši porabi virov in hitrejšemu zagonu, kot če bi uporabili navidezni računalnik. Docker sem uporabil, ker je z njim zelo enostavno postaviti strežnik Zanata [2].

2.4 Sublime Text

Sublime Text je urejevalnik besedila, ki sem ga uporabil za razvoj svoje aplikacije. Uporabil sem ga, ker mi je všeč nabor barv, ki jih uporablja, kar naredi kodo lepo berljivo in ker je hiter in enostaven za uporabo. Podpira tudi veliko število programskih in označevalnih jezikov, poleg tega se programu lahko dodajo dodatne funkcionalnosti preko vtičnikov.



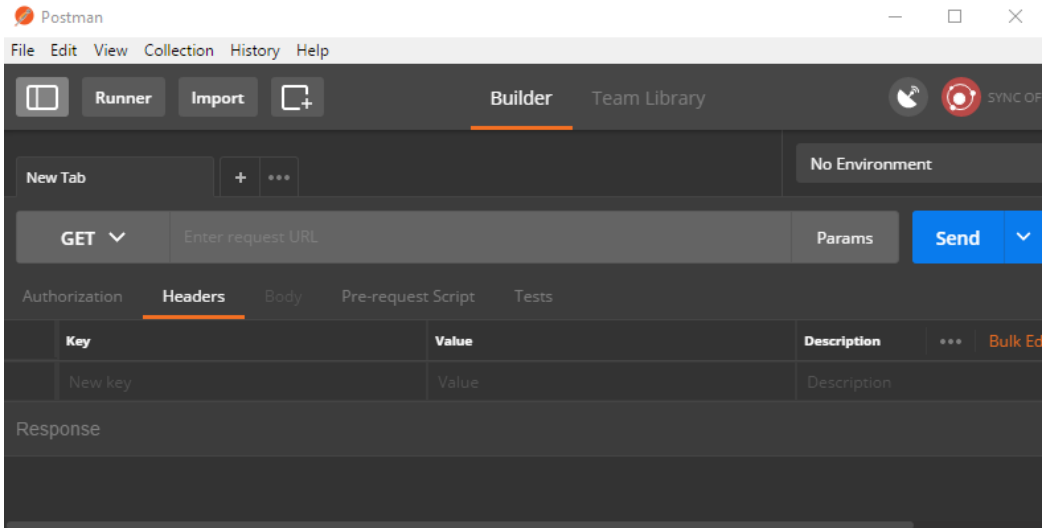
Slika 2.3: Primerjava Docker arhitekture z arhitekturo navideznih računalnikov [7]



Slika 2.4: Sublime Text urejevalnik [8]

2.5 Postman

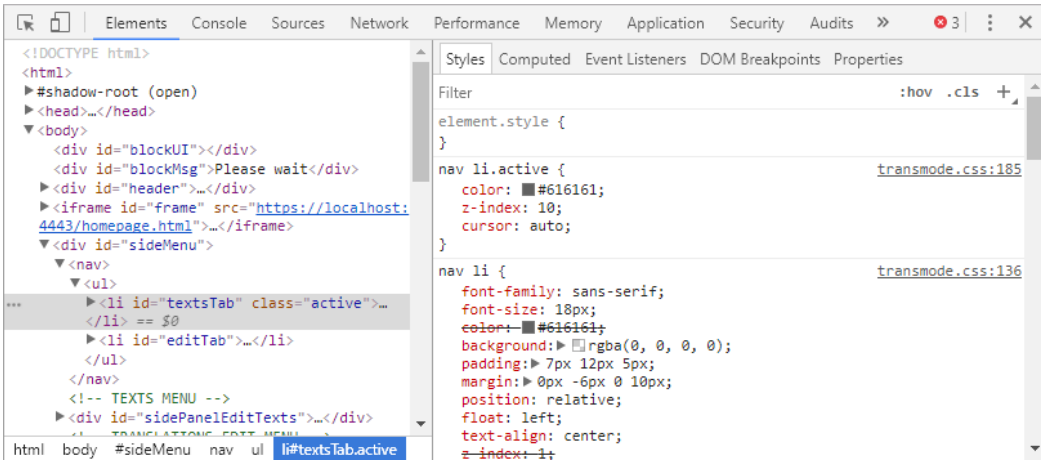
Postman je program, s katerim lahko enostavno izvajamo HTTP zahtevke. Nastavljamo lahko različne parametre zahtevka, kot so URL, glavo zahtevka, telo zahtevka in drugo. Uporabil sem ga predvsem za testiranje Zanata REST API, kar mi je prišlo zelo prav, saj sem potem vedel kako je zgrajen uspešen zahtevek in mi je olajšalo delo pri pisanju zahtevkov v JavaScriptu. Zelo uporabna je tudi konzola, ki izpiše status poslanega zahtevka in morebitne napake.



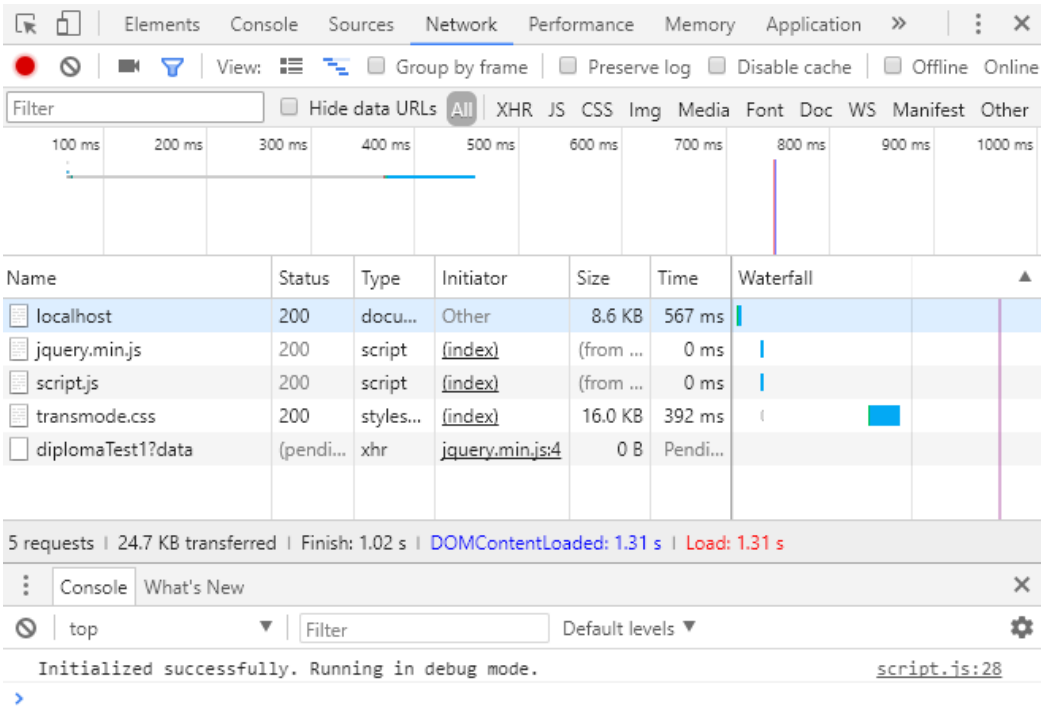
Slika 2.5: Program Postman

2.6 Chrome

Za testiranje in razhroščevanje svoje aplikacije sem uporabil brskalnik Chrome. Je hiter, odziven in pregleden. Večinoma sem ga uporabljal v načinu za pregled, ki ima veliko uporabnih funkcionalnosti, kot so pregled in spreminjanje izvorne kode, konzolo, kjer se izpišejo napake, možnost razhroščevanja, pregled zahtevkov in drugo.



Slika 2.6: Pregled HTML in CSS kode v brskalniku Chrome



Slika 2.7: Pregled zahtevkov v brskalniku Chrome

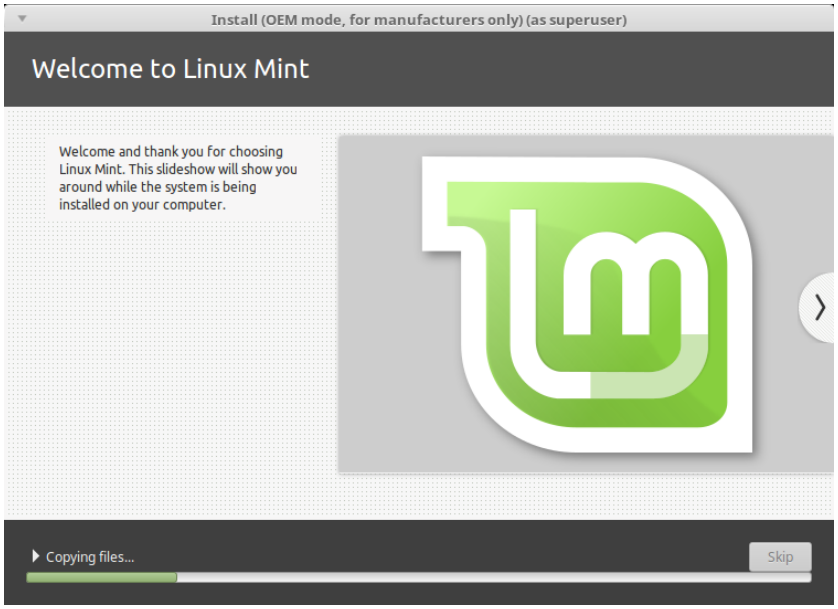
Poglavje 3

Postavitev in konfiguracija orodja Zanata

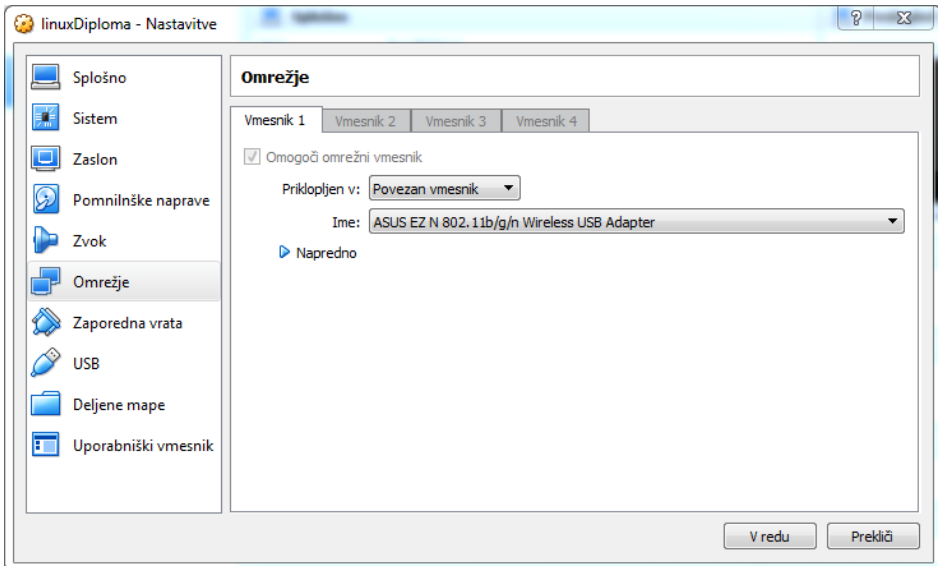
V tem poglavju bom predstavil konfiguracijo in orodja, ki jih potrebujemo za postavitev strežnika Zanata. Pokazal bom tudi kako pripravimo projekt v Zanati in kako pridobimo avtentikacijske podatke, ki jih potrebujemo za izvedbo REST klicev.

3.1 Virtualbox in Linux

Razvijalci orodja Zanata so pripravili skripte za operacijski sistem Linux, s katerimi z uporabo orodja Docker enostavno zaženemo strežnik Zanata. Ker sem razvijal na računalniku z nameščenim operacijskim sistemom Windows, sem uporabil Virtualbox, da sem lahko poganjal Linux. Odločil sem se za uporabo Linux Mint različice, ker je ena izmed bolj uporabniku prijaznih različic. Sliko za namestitev Linuxa sem prenesel preko interneta [3] in jo uporabil za namestitev v Virtualboxu. Za pravilno delovanje interneta sem v nastavitvah Virtualboxa nastavil omrežni vmesnik na način **povezan vmesnik**.



Slika 3.1: Namestitev operacijskega sistema Linux Minut v Virtualboxu



Slika 3.2: Omrežne nastavitve v Virtualbox-u

3.2 Docker

Docker na operacijskem sistemu Linux Mint zelo enostavno namestimo z naslednjim ukazom.

```
sudo apt-get install docker.io
```

Po uspešni namestitvi potrebujemo skripte, ki ustvarijo Docker sliko za strežnik Zanata in za zagon strežnika. Skripte so že pripravili razvijalci Zanate, so odprtokodne in objavljene na [11]. Po prenosu skript, se pomaknemo v direktorij s skriptami in ustvarimo Docker sliko, ki bo vsebovala najnovejšo različico strežnika Zanata s spodnjim ukazom.

```
sudo docker build -t zanata/server .
```

Ko se ukaz zaključi, lahko strežnik enostavno zaženemo v Docker vsebniku z uporabo že pripravljene skripte.

```
sudo ./runapp.sh -e <email-naslov>
```

Po uspešnem zagonu strežnika, lahko do njega dostopamo v brskalniku na spodnjem naslovu.

```
http://localhost:8080/zanata
```

Za poln nabor funkcionalnosti potrebujemo uporabnika z administratorskimi pravicami. S spodnjimi ukazi lahko ustvarimo administratorja z uporabniškim imenom "admin"in geslom "admin1234".

```
sudo apt install mysql-client-core-5.7

DB_IP=$(docker inspect --format
'{{range .NetworkSettings.Networks}} {{.IPAddress}}{{end}}'
zanatadb)

mysql --protocol=tcp -h $DB_IP -u zanata -p zanata
< conf/admin-user-setup.sql
```

3.3 Priprava projekta v Zanati

Strežniška aplikacija Zanata omogoča enostavno ustvarjanje novih projektov. Vsak projekt ima lahko več različic, slednje pa več dokumentov. Pri ustvarjanju različice, lahko nastavimo v kakšni obliki želimo imeti dokumente. Izberemo lahko navadno datoteko, kar nam omogoči, da ustvarimo dokument tako, da izberemo datoteko na računalniku. Sam sem uporabil datoteko s končnico **properties**. V takem primeru mora datoteka biti strukturirana na sledeč način:

```
ključ=izvorna vrednost besedila (običajno angleški prevod)
ključ2=izvorna vrednost besedila (običajno angleški prevod)
.
.
.
```

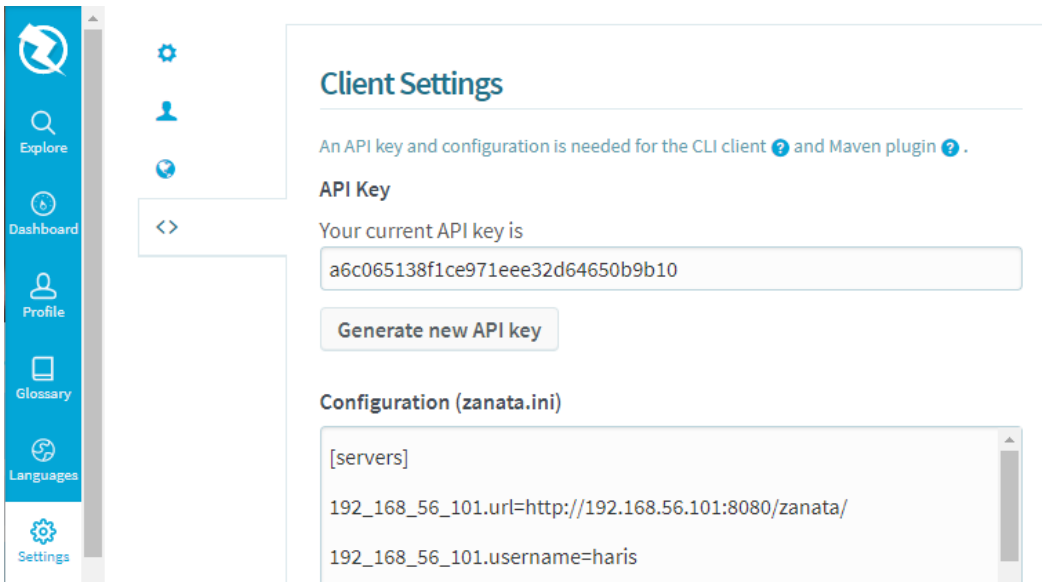
Po uspešnem prenosu datoteke na strežnik, se nam ustvari dokument z enakim imenom, kot je bilo ime datoteke. Dokument bo imel toliko jezikov, kot jih je nastavljenih na projektu. Ko odpremo dokument za določen jezik, se nam na levi strani za vsak ključ v dokumentu prikaže izvorno besedilo, na desni strani pa lahko ključu nastavimo vrednost oziroma prevod za izbrani jezik.

3.4 Avtentikacija

Zanata uporablja uporabniško ime in API ključ za avtentikacijo uporabnika pri REST klicih. Če imamo ustrezne pravice in se uspešno avtenticiramo, bodo naši REST klici uspešni. Avtentikacija je uporabljena tudi za funkcionalnost vodenja evidenc sprememb prevodov.

3.4.1 Kako dobimo API ključ

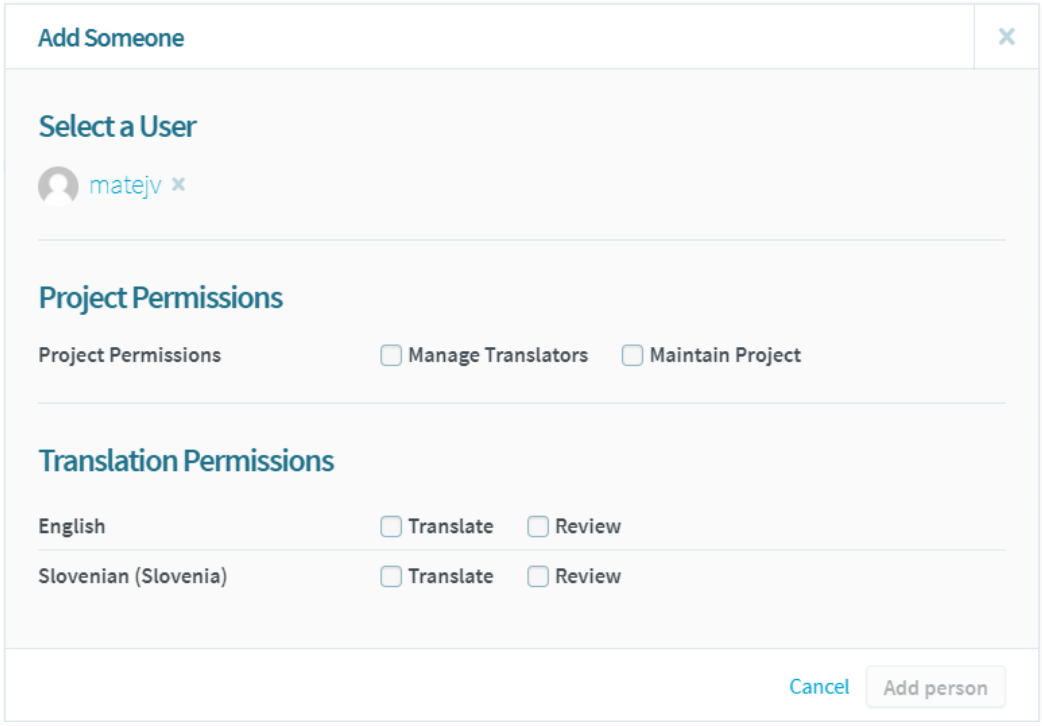
Po uspešni prijavi v spletni aplikaciji Zanata, si lahko v uporabniških nastavitvah generiramo svoj API ključ.



Slika 3.3: Generiranje API ključa za strežniško aplikacijo Zanata

3.4.2 Kako uporabniku dodelimo pravice na projektu

Da lahko uporabnik ureja prevode, mora imeti tudi ustrezne pravice na posameznem projektu, katere mu dodeli administrator v Zanati. Uporabnik brez pravic s svojimi avtentikacijskimi podatki tudi ne more uspešno izvesti REST klicev na strežnik Zanata. Pravice uporabniku dodelimo tako, da odpremo projekt, kliknemo gumb **add user**, poiščemo uporabnika in ga ustrezno pooblastimo.



Slika 3.4: Dodajanje pravic za prevajanje uporabniku matejv

Poglavje 4

Opis grafičnega uporabniškega vmesnika

V tem poglavju je opisan GUI za prevajanje spletnih aplikacij. Predstavljeno je, kako se ga doda v spletno aplikacijo, opisane so funkcionalnosti in problemi na katere sem naletel med izdelavo.

4.1 Struktura projekta in način uporabe

Projekt je sestavljen iz treh datotek:

- **translation_gui.html** - je glavna datoteka, ki vsebuje html kodo za strukturo vmesnika in JavaScript kodo za logiko. Preostali dve datoteki sta uporabljeni znotraj te datoteke.

- **translation_gui.css** - je datoteka s css kodo za GUI.

- **zanata.js** - je datoteka, ki vsebuje JavaScript kodo za delo z strežnikom Zanata. Vključuje vse REST klice na strežnik Zanata in strukture, ki hranijo podatke o projektu in prevodih.

Za uporabo GUI aplikacije obstajata dva predpogoja in sicer:

1. Potrebujemo strežnik Zanata, ki vsebuje projekt z enoličnimi oznakami oz. ključi za tekste, ki jih uporablja naša spletna aplikacija.

2. Spletna aplikacija mora biti zasnovana tako, da ima vsak tekst, ki ga želimo prevajati nastavljen data atribut z vrednostjo ključa.

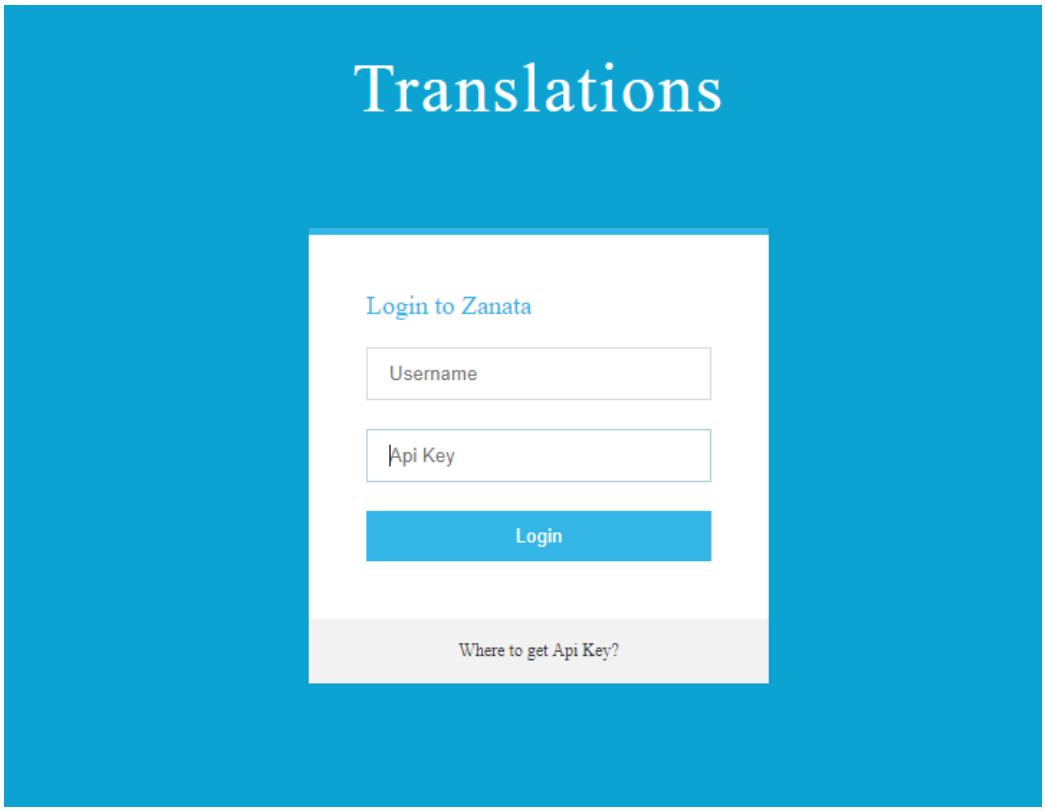
GUI vključimo v spletno aplikacijo tako, da dodamo vse tri datoteke in ustrezno popravimo datoteko **translation_gui.html**. To storimo tako, da popravimo parametre za inicializacijo.

```
$( document ).ready(function() {
  zanata.init({
    serverUrl: "https://192.168.1.137:8443/zanata",
    iframeUrl: "https://localhost:4443/webapp.html",
    project: "diplomaTest1",
    versions: ["version1"],
    dataKeyId: "transkey"
  });
});
```

Opis parametrov:

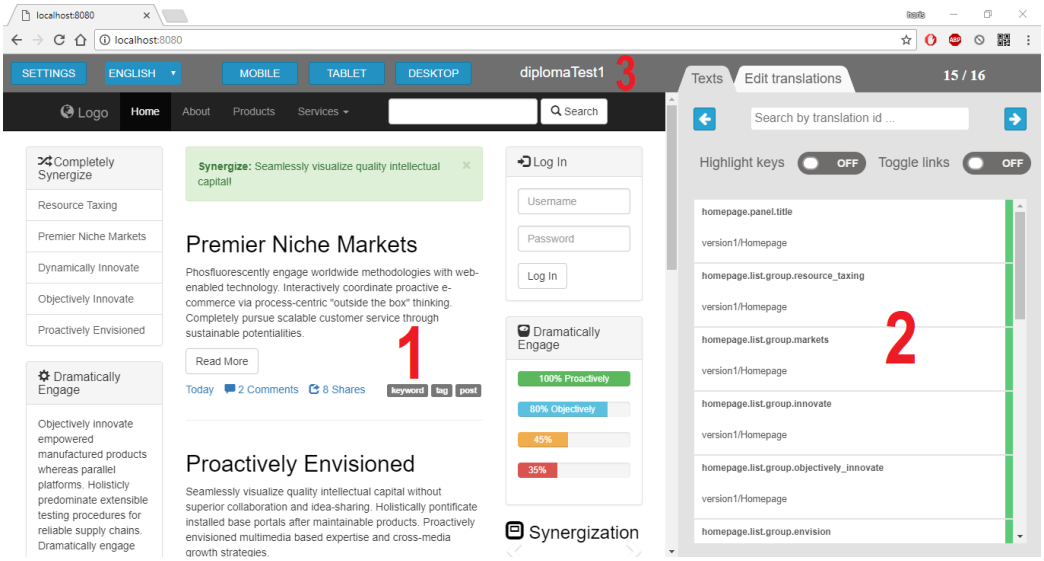
- **serverUrl** - je URL do strežnika Zanata
- **iframeUrl** - je URL do spletne aplikacije, ki jo želimo prevajati.
- **project** - je ime projekta v spletni aplikaciji Zanata, ki hrani prevode naše aplikacije.
- **versions** - je seznam različic prevodov, ki jih uporabljamo
- **dataKeyId** - je ime data atributa v naši spletni aplikaciji, ki nosi informacijo o ključu za posamezen tekst v spletni aplikaciji.

Ko imamo vse prav nastavljeno lahko začnemo uporabljati uporabniški vmesnik. To storimo tako, da enostavno v brskalniku odpremo datoteko **translation_gui.html**. Prikazala se nam bo login stran, kjer se bo uporabnik prijavil z uporabniškim imenom in API ključem za strežnik Zanata. Za login stran je uporabljena zastonjska predloga iz interneta [4], ki je bila malo predelana.



Slika 4.1: Stran za prijavo v uporabniški vmesnik

Po uspešni prijavi, se bodo iz strežnika Zanata prenesli vsi prevodi in na desni strani se nam bodo v vmesniku prikazali vsi ključi, ki se uporabljajo na trenutni spletni strani, ter status o prevodu.



Slika 4.2: GUI po uspešni prijavi

4.2 Zgradba in funkcionalnosti GUI

Ko razvijamo, posvečamo večino pozornosti delovanju aplikacije. Čeprav igra uporabniška izkušnja pomembno vlogo pri uspehu izdelka je pogosto zanemarjena. Zato, da bi bila uporabniška izkušnja čim boljša je GUI zelo preprost [14]. Sestavljen je iz treh komponent. Komponente so označene na sliki s številkami. To so:

1. Komponenta številka 1 je HTML iframe element. To je okno, ki naloži spletno aplikacijo, katero trenutno prevajamo.

2. Komponenta številka 2 je stranski meni z dvema zavihkoma.

3. Komponenta številka 3 je orodna vrstica z gumbi in podatki o projektu.

4.2.1 Iframe

Glavna funkcija komponente z iframe elementom je, da naloži in prikaže spletno aplikacijo, ki jo želimo prevajati. Poleg tega GUI omogoča tudi to,

da na tekst znotraj iframe elementa, ki ima pravilno nastavljen data atribut enostavno kliknemo in se nam v stranskem meniju odpre zavihek za urejanje prevodov. Tako lahko enostavno in hitro prevajamo besedila s klikanjem po aplikaciji.

4.2.2 Stranski meni

Stranski meni je sestavljen iz dveh zavihkov. Oba zavihka imata naslednje funkcionalnosti:

- dva gumba z uporabo katerih se lahko enostavno sprehajamo po ključih na spletni strani.

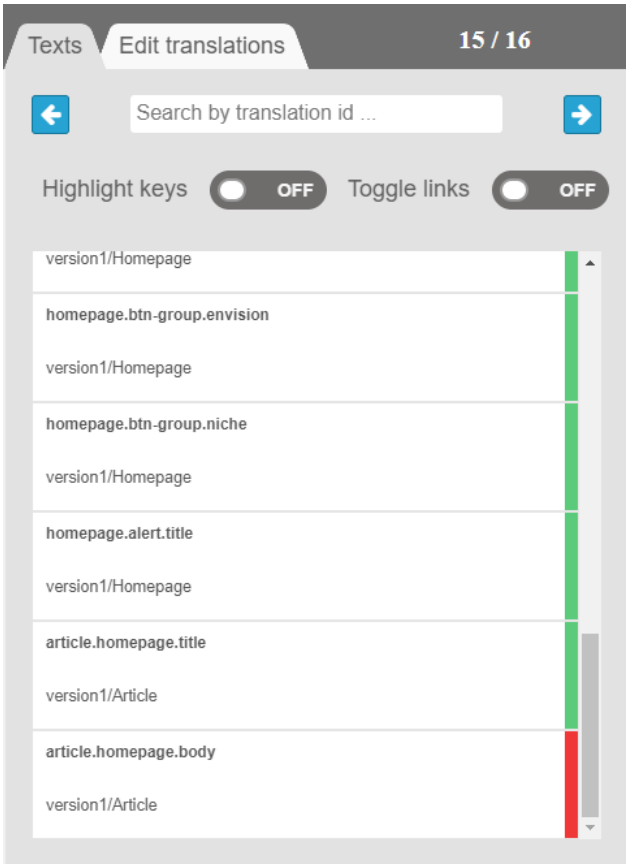
- iskalnik ključa, s katerim lahko filtriramo ključe na spletni strani. Iščemo lahko tudi samo po delu ključa. Tukaj ima GUI veliko prednost pred prevajanjem v spletni aplikaciji Zanata, saj le ta ne podpira iskanje po le delu ključa.

Zavihek **Texts** ima poleg osnovnih funkcij še:

- gumb, ki označi vsa besedila, katera je možno prevajati

- gumb, ki onemogoči povezave na strani. Tako ne pride do situacije, ko kliknemo na povezavo, ker želimo prevesti tekst in nas nato aplikacija preusmeri.

- okno, ki prikaže vse ključe, ki se uporabljajo na strani. Poleg tega se pod vsakim ključem izpiše v kateri različici in v katerem dokumentu se nahaja v spletni aplikaciji Zanata. Na desni strani posamezne vrstice se prikaže tudi barvna oznaka, ki označuje status prevoda v aplikaciji Zanata za trenutno izbran jezik.

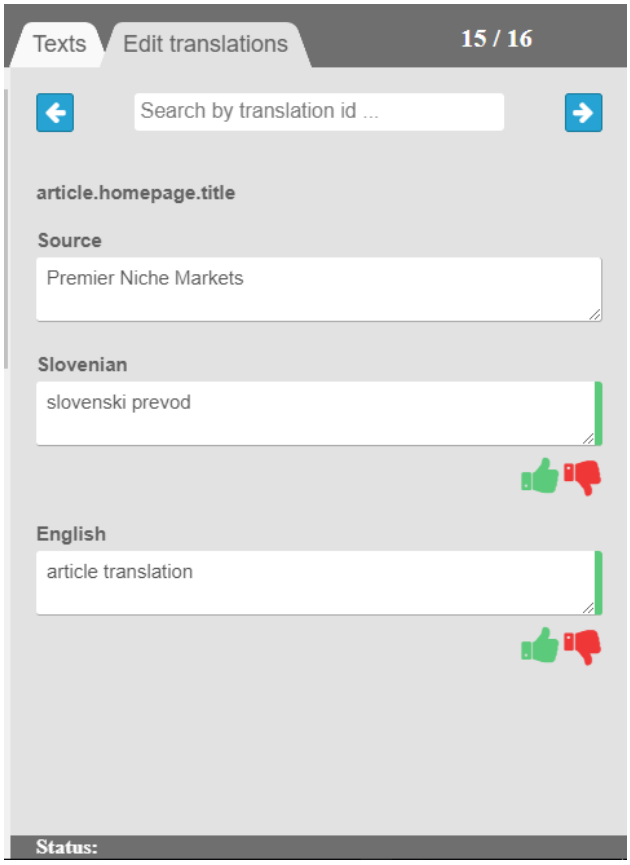


Slika 4.3: Zavihek **Texts** v stranskem meniju uporabniškega vmesnika

Zavihek **Edit translations** omogoča naslednje:

- izpis imena ključa, katerega trenutno urejamo.
- izpis izvornega besedila, kot je nastavljeno v spletni aplikaciji Zanata.
- izpis prevodov za ključ v vseh jezikih, ki so nastavljeni na projektu v spletni aplikaciji Zanata.
- spreminjanje prevodov za posamezen jezik. Ko spremenimo prevod za posamezen jezik in potrdimo spremembo z zelenim gumbom, se bo vrednost v spletni aplikaciji Zanata ustrezno spremenila in nastavila

status prevoda kot preveden. Če uporabimo rdeč gumb, se bo status v spletni aplikaciji Zanata za ta ključ nastavil na zavrnjeno.



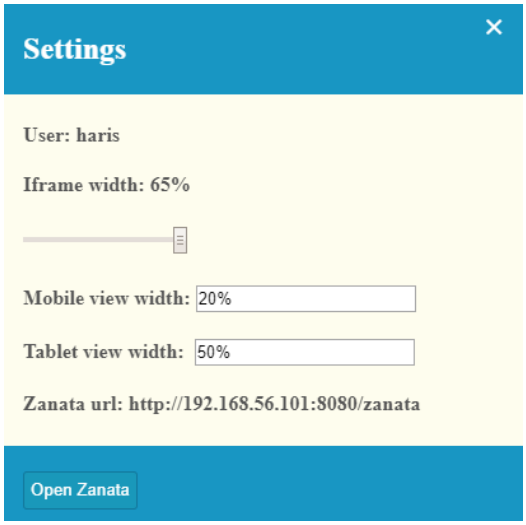
Slika 4.4: Zavihek **Edit translations** v stranskem meniju uporabniškega vmesnika

4.2.3 Orodna vrstica

Orodna vrstica se uporablja za različne nastavitve in prikaz podatkov o projektu. Omogoča naslednje:

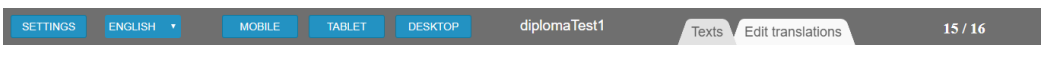
- gumb s katerim odpremo okno z nastavitvami. V nastavitvah se izpiše uporabniško ime, s katerim smo se prijavili in URL do strežnika Zanata.

Urejamo lahko širino iframe elementa in imamo gumb, ki odpre spletno aplikacijo Zanata v novem zavihku.



Slika 4.5: Dodatne nastavitve

- gumb s katerim izberemo jezik spletne strani
- ima tri gumbe, ki spreminjajo širino iframe elementa in s tem dosežejo prikaz spletne strani, kot bi se prikazala na različnih napravah.
- prikaz imena projekta, ki ga urejamo
- prikaz koliko ključev od vseh je prevedenih na strani



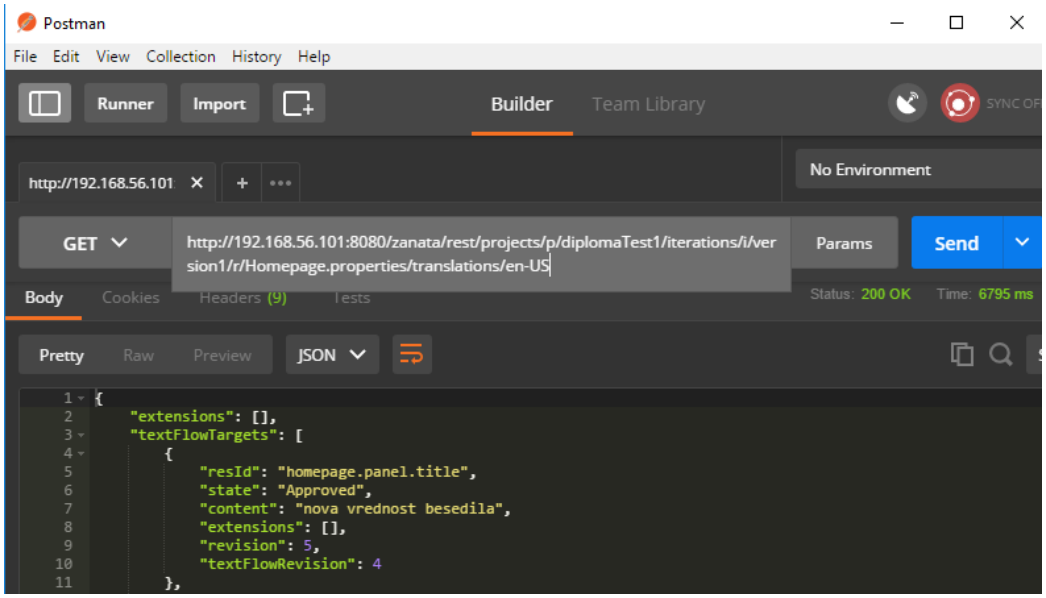
Slika 4.6: Orodna vrstica GUI

4.3 Primeri rest klicev

Dokumentacija REST API klicev za strežnik Zanata je javno objavljena [13]. Klice lahko enostavno testiramo z uporabo Postman aplikacije. V glavi zahtevka lahko povemo v kakšni obliki želimo odgovor. Polega tega v glavi nastavimo parametre z našimi avtentikacijskimi podatki. Brez pravilnih avtentikacijskih podatkov, bomo dobili odgovor s statusom 401. To pomeni, da nismo pooblaščeni za izvedbo zahtevka.

4.3.1 HTTP GET

Primer HTTP GET klica za pridobitev podatkov o vseh ključih v določenem dokumentu na strežniku Zanata.



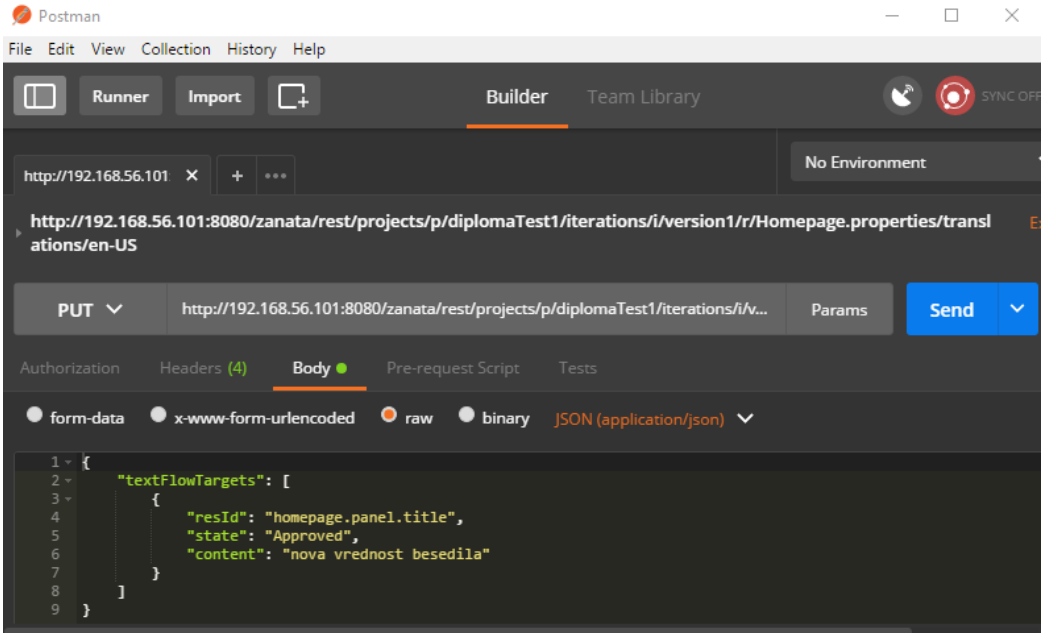
Slika 4.7: Primer izvedbe HTTP GET zahtevka v Postman aplikaciji

```
var getDocumentSources = function(documentName,callback){
  $.ajax({
        headers: {
            "x-auth-user":  zanata.username,
            "x-auth-token": zanata.apiKey,
            "Accept" : "application/json; charset=utf-8",
            "Content-Type": "application/json;charset=utf-8"
        },
        url: zanata.url+"/rest/projects/p/"+zanata.project+"/iterations/i/"+obj.version+"/r/"+documentName,
        type: "GET",
        data: "data",
        cache: false,
        success : function(data){
          for(indeks in data["textFlows"]){
              var resource = data["textFlows"][indeks];
              obj.sources[documentName][resource["id"]] = resource["content"];
          }
          callback();
        }
  })
}
```

Slika 4.8: Implementacija HTTP GET zahtevka v kodi

4.3.2 HTTP PUT

Primer HTTP PUT zahtevka za spreminjanje vrednosti ključa za določen dokument in jezik.



Slika 4.9: Primer izvedbe HTTP PUT zahtevka v Postman aplikaciji

```
var putTranslation = function(key, translation, state, doc, lang, callback){
  data = JSON.stringify(prepareJsonTrans(key,translation,state));
  $.ajax({
    headers: {
        "Accept" : "application/json; charset=utf-8",
        "Content-Type": "application/json;charset=utf-8",
        "x-auth-user": zanata.username,
        "x-auth-token": zanata.apiKey
    },
    url: zanata.url+"/rest/projects/p/"+zanata.project+"/iterations/i/"+obj.version+"/r/"+doc+"/translations/"+lang,
    type: "PUT",
    data: data,
    complete : function(xhr, textStatus){
      var success = xhr.status == "200" ? true : false;
      if(success){
        var newtrans = {};
        newtrans["content"] = trans;
        newtrans["state"] = state;
        obj.translations[lang][key] = newtrans;
      }
      callback(success,xhr,lang,key,trans,state);
    }
  })
}
```

Slika 4.10: Implementacija HTTP PUT zahtevka v kodi

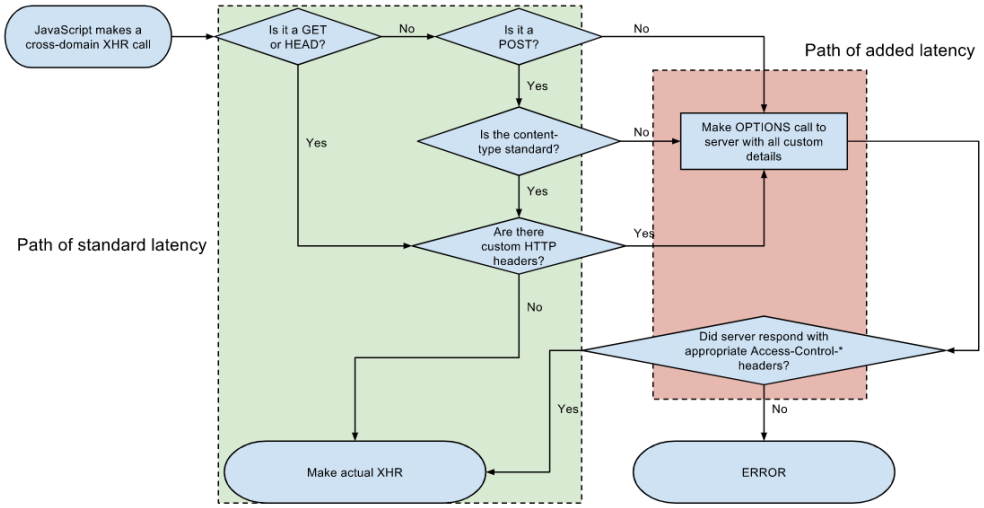
4.4 Problemi

V nadaljevanju bom opisal različne probleme, ki sem jih imel med izdelavo aplikacije in načine kako sem jih rešil.

4.4.1 CORS

Je mehanizem, ki definira način obnašanja strežnika in brskalnika, ko pride do zahtevka, ki je prišel iz različne domene kot je strežniška domena. Z uporabo CORS imamo več svobode, kot če bi na strežniku dovolili samo zahtevke, ki so prišli iz iste domene in večjo varnost, kot če bi dovolili vse zahtevke, ki prihajajo iz različnih domen [1].

Med izdelavo aplikacije, sem opazil, da strežnik Zanata za določene REST klice ne dovoljuje CORS zahtevkov. Preko elektronske pošte sem poslal sporočilo razvijalcem Zanate, ki so kasneje dodali nastavitev v konfiguracijski datoteki, kjer lahko nastavimo iz katerih domen strežnik Zanata dovoljuje REST klice.



Slika 4.11: Diagram prikazuje delovanje brskalnika pri pošiljanju CORS zahtevka [6]

Nastavitev dovoljenih domen:

```
<property name="zanata.origin.whitelist"
value="http://localhost:8080" />
```

Z zgornjo nastavitvijo, bo strežnik Zanata dovolil zahtevke, ki pridejo iz domene localhost z vrati 8080. Da sem lahko pošiljal REST klice na strežnik Zanata, sem si moral postaviti HTTP strežnik, ki bo poslušal na vratih 8080. Uporabil sem zelo preprost python ukaz prikazan na spodnji sliki.

```
C:\Users\Haris\Desktop\server>python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Slika 4.12: Prikaz izvedbe python ukaza, ki zažene preprost HTTP strežnik

Lahko uporabimo tudi ukaz, ki požene brskalnik Chrome v načinu, kjer ne preverja, če gre za CORS zahtevek. Na ta način sem lahko razvijal naprej, čeprav razvijalci Zanate še niso dodali nove funkcionalnosti.

```
C:\Users\Haris\Desktop\server>"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe"
--disable-web-security --user-data-dir="c:/chromedev"
```

Slika 4.13: Ukaz, ki zažene brskalnik Chrome v načinu brez varnostnih mehanizmov

4.4.2 Stran z mešano vsebino

Stran z mešano vsebino (ang. mixed content) je spletna stran, ki uporablja protokol HTTPS, hkrati pa nekatere elemente strani naloži preko protokola HTTP. Take strani so ranljive, saj so le delno enkriptirane. Pri izdelavi diplomske naloge, sem naletel na ta problem, ko sem na strani, ki uporablja protokol HTTPS želel klicati REST API Zanata strežnika preko HTTP protokola [5].

```
▶ Mixed Content: The page at                                    xmlhttprequest-example.html:38
'https://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/xmlhttprequest-example.html' was loaded over
HTTPS, but requested an insecure XMLHttpRequest endpoint
'http://googlesamples.github.io/web-fundamentals/samples/discovery-and-
distribution/avoid-mixed-content/xmlhttprequest-data.js'. This request has been
blocked; the content must be served over HTTPS.
```

Slika 4.14: Prikaz napake v konzoli brskalnika, ko izvedemo HTTP klic na HTTPS strani

4.4.2.1 HTTPS na strežniku Zanta

Problem s stranjo z mešano vsebino, sem rešil tako, da sem omogočil HTTPS protokol na strežniku Zanata in klice na strežnik Zanata ustrezno spremenil. HTTPS sem omogočil na sledeč način:

1. s pomočjo pomožnega orodja keytool sem generiral datoteko z zasebnim ključem z naslednjim ukazom:

   ```
   keytool -genkey -alias mycert -keyalg RSA -sigalg
   MD5withRSA -keystore ./zanata.keystore -storepass
   ```

```
123456 -dname "cn=haris"
-keypass 123456 -validity 9999
```

2. v konfiguracijski datoteki strežnika Zanata sem omogočil HTTPS. Definiral sem naslednjo konfiguracijo, kjer sem med drugim nastavil pot do datoteke, ki sem jo generiral v prvem koraku.

```
<security-realm name="UndertowRealm">
  <server-identities>
    <ssl>
      <keystore path="${jboss.server.config.dir}
        /zanata.keystore" keystore-password="123456"
        alias="mycert" key-password="123456"/>
    </ssl>
  </server-identities>
</security-realm>
```

Nato sem v isti datoteki nastavil, da strežnik za HTTPS uporabi zgornje nastavitve.

```
<https-listener name="https" secure="true"
security-realm="UndertowRealm" socket-binding="https"/>
```

3. popravil sem Docker skripte, tako da sem izpostavil vrata 8443, na katerih strežnik Zanata posluša za HTTPS zahtevke. Poleg tega sem dodal ukaz, ki doda datoteko z zasebnim ključem pri gradnji Zanata Docker slike.

```
EXPOSE 8443
ADD conf/zanata.keystore
/opt/jboss/wildfly/standalone/configuration/
```

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge je bila narejena JavaScript knjižnica za delo s strežnikom Zanata, glavni prispevek pa je GUI, ki z uporabo knjižnice olajša prevajanje spletnih aplikacij.

Podjetja in razvijalci, ki že koristijo strežniško aplikacijo Zanata in želijo uporabljati strežnikov REST API, lahko uporabijo razvito knjižnico in jo prilagodijo za svoje potrebe. Poleg tega se lahko odločijo tudi za uporabo GUI aplikacije, s katerim še dodatno pohitrijo razvojni proces. Vmesnik lahko vključimo v katerokoli spletno aplikacijo, saj uporablja standardne spletne tehnologije.

Vmesnik bi lahko še izboljšali tako, da bi dodali funkcionalnost avtomatskega prevajanja. Lahko bi na primer uporabili kakšen spletni servis za prevajanje kot je Google prevajalnik.

Mislim, da je cilj diplomske naloge dosežen, saj v podjetju vmesnik že uporabljajo. Uporabniki so zadovoljni s preprosto uporabo in izboljšanim iskanjem ključev v primerjavi s strežnikom Zanata.

Literatura

[1] Cors. Dosegljivo: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing. [Dostopano 20. 6. 2017].

[2] Docker. Dosegljivo: https://en.wikipedia.org/wiki/Docker_(software). [Dostopano 24. 12. 2017].

[3] Linux mint. Dosegljivo: https://linuxmint.com/download.php. [Dostopano 24. 12. 2017].

[4] Login predloga. Dosegljivo: https://dcrazed.com/css-html-login-form-templates/. [Dostopano 7. 7. 2017].

[5] Mešana vsebina. Dosegljivo: https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content. [Dostopano 21. 6. 2017].

[6] Slika cors zahtevka. Dosegljivo: https://en.wikipedia.org/wiki/File:Flowchart_showing_Simple_and_Preflight_XHR.svg. [Dostopano 20. 6. 2017].

[7] Slika primerjava dockerja z navideznim računalnikom. Dosegljivo: https://zdnet2.cbsistatic.com/hub/i/r/2017/05/08/af178c5a-64dd-4900-8447-3abd739757e3/resize/770xauto/78abd09a8d41c182a28118ac0465c914/docker-vm-container.png. [Dostopano 24. 12. 2017].

[8] Slika programa sublime text. Dosegljivo: https://i.ytimg.com/vi/BAOCOR4lzEg/maxresdefault.jpg. [Dostopano 15. 6. 2017].

[9] Virtualbox. Dosegljivo: https://en.wikipedia.org/wiki/VirtualBox. [Dostopano 24. 12. 2017].

[10] Zanata aplikacija. Dosegljivo: https://translate.zanata.org/. [Dostopano 24. 12. 2017].

[11] Zanata docker github. Dosegljivo: https://github.com/zanata/zanata-docker-files/tree/master/zanata-server. [Dostopano 24. 12. 2017].

[12] Zanata github. Dosegljivo: https://github.com/zanata/zanata-platform. [Dostopano 15. 6. 2017].

[13] Zanata rest dokumentacija. Dosegljivo: https://zanata.ci.cloudbees.com/job/zanata-api-site/site/zanata-common-api/rest-api-docs/index.html. [Dostopano 15. 6. 2017].

[14] Jesse James Garrett. *The elements of user experience: user-centered design for the web and beyond.* Pearson Education, 2010.